Fichier
scr/graphics/img/icon.xbm

```
#ifndef USERPREFS_HAS_SPLASH
#define icon_width 50
#define icon_height 28
static uint8_t icon_bits[] = {
  0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x01, 0x38, 0x00, 0xFC, 0xFF, 0x03, 0x00, 0x70, 0x00, 0xFE,
  0xFF, 0xFF, 0x7F, 0x00, 0x08, 0x05, 0xFE, 0xFF, 0xFF, 0x7F, 0x00, 0x08,
  0xC0, 0xFF, 0xDF, 0x80, 0x7F, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x02, 0x00,
  0x00, 0x30, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00,
  0x12, 0x00, 0x00, 0x80, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
  0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x0F, 0x00, 0x00, 0x80, 0x01, 0x00, 0x80, 0xFF, 0x00, 0x00, 0x00,
  0xFF, 0x00, 0x80, 0x1F, 0x1C, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x3F, 0x00,
  0x04, 0x80, 0xFF, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x80, 0xFE, 0x00, 0x00,
  0xFF, 0x01, 0x00, 0x40, 0xFE, 0x00, 0x00, 0xFE, 0x0F, 0x00, 0x20, 0xFE,
  0x00, 0x00, 0xFC, 0x0F, 0xFA, 0x01, 0xFE, 0x00, 0x00, 0xF8, 0x3F, 0x00,
  0x00, 0xFE, 0x00, 0x00, 0xE0, 0xFF, 0x01, 0x00, 0xFE, 0x00, 0x00, 0xC0,
  0xFF, 0x1F, 0xC0, 0xFF, 0x00, 0x00, 0x00, 0xFE, 0xFF, 0xFF, 0xFF, 0x00,
  0x00, 0x00, 0xE0, 0xFF, 0xFF, 0x7F, 0x00, 0x00, 0x00, 0x00, 0xFC, 0xFF,
  0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, };
#endif
```

scr/graphics/Screen.cpp

ligne 58

```
#if HAS_ETHERNET
#include "mesh/eth/ethClient.h"
#endif
```

ligne 169

```
display->setFont(FONT_MEDIUM);
display->setTextAlignment(TEXT_ALIGN_LEFT);
const char *title = "gaulix.fr";
display->drawString(x + getStringCenteredX(title), y + SCREEN_HEIGHT -
FONT_HEIGHT_MEDIUM, title);
display->setFont(FONT_SMALL);
```

ligne 238

```
#if HAS_ETHERNET
static void drawEthernetFrame(OLEDDisplay *display, OLEDDisplayUiState *state, int16_t
x, int16_t y)
{
display->setFont(FONT_SMALL);
display->setTextAlignment(TEXT_ALIGN_LEFT);
```

```
if (config.display.displaymode ==
meshtastic_Config_DisplayConfig_DisplayMode_INVERTED) {
display->fillRect(0 + x, 0 + y, x + display->getWidth(), y + FONT_HEIGHT_SMALL);
display->setColor(BLACK);
}

display->setColor(WHITE);
// Position verticale ajustée - commence plus haut
int16_t y_offset = y + 2; // Réduit l'espace au-dessus
// Alignement à gauche (x + petit offset)
int16_t x_offset = x + 2;
// Affichage non centré, aligné à gauche
display->drawString(x_offset, y_offset, "Ethernet Config:");
y_offset += FONT_HEIGHT_SMALL + 2; // Espacement légèrement réduit
display->drawString(x_offset, y_offset, "IP: " + Ethernet.localIP().toString());
y_offset += FONT_HEIGHT_SMALL;
display->drawString(x_offset, y_offset, "Mask: " + Ethernet.subnetMask().toString());
y_offset += FONT_HEIGHT_SMALL;
display->drawString(x_offset, y_offset, "GW: " + Ethernet.gatewayIP().toString());
//y_offset += FONT_HEIGHT_SMALL;
//display->drawString(x_offset, y_offset, "DNS: " + Ethernet.dnsServerIP().toString());
}
#endif
```

ligne 453

```
snprintf(tempBuf, sizeof(tempBuf), "Critical fault #%d", error_code);
display->drawString(0 + x, 0 + y, tempBuf);
display->setTextAlignment(TEXT_ALIGN_LEFT);
display->setFont(FONT_SMALL);
display->drawString(0 + x, FONT_HEIGHT_MEDIUM + y, "For help, please visit \ngaulix.fr");
}
```

ligne 2215

```
#if HAS_ETHERNET
if (Ethernet.hardwareStatus() != EthernetNoHardware) {
normalFrames[numframes++] = drawEthernetFrame;
}
#endif
```

scr/mesh/RadioInterface.cpp

ligne 48

```
RDEF(EU_868, 869.4f, 869.65f, 10, 0, 27, false, false, false),
//RDEF(EU_868, 869.4f, 869.65f, 10, 0, 33, false, false, false),
```

scr/mesh/api/serverapi.h
ligne 47

```
//#if RAK_4631
#if defined(RAK_4631) || defined(RAK11310)
// Track wait time for RAK13800 Ethernet requests
int32_t waitTime = 100;
#endif
```

scr/mesh/eth/ethClient.cpp
ligne 115

```
#ifdef RAK11310 // Initialize the SPI port
ETH_SPI_PORT.setSCK(PIN_SPI0_SCK);
ETH_SPI_PORT.setTX(PIN_SPI0_MOSI);
ETH_SPI_PORT.setRX(PIN_SPI0_MISO);
ETH_SPI_PORT.begin();
#endif
Ethernet.init(ETH_SPI_PORT, PIN_ETHERNET_SS);

uint8_t mac[6];
```

dans
scr/modules  on ajoute 2 fichiers
NivekRemoteModule.cpp

```
#include "NivekRemoteModule.h"
#include "configuration.h"
#include "MeshService.h"

bool NivekRemoteModule::wantPacket(const meshtastic_MeshPacket *p) {
if (MeshService::isTextPayload(p) && p->decoded.payload.size >= 12) { // 9 (Node ID) + 2
(Pin) + 1 (State)
// Récupérer l'ID du nœud en tant que chaîne de caractères
NodeNum nodeID = myNodeInfo.my_node_num;
char nodeIDStr[10]; // Stocke l'ID en tant que string
snprintf(nodeIDStr, sizeof(nodeIDStr), "%u", nodeID); // Convertir en string
// Comparer les 9 premiers caractères avec l'ID du nœud
if (strncmp((char*)p->decoded.payload.bytes, nodeIDStr, 9) == 0) {
// Extraire le numéro de sortie (2 caractères après l'ID du nœud)
int pin = ((p->decoded.payload.bytes[9] - '0') * 10) + (p->decoded.payload.bytes[10] -
'0');

// Vérifier que la valeur est bien entre 0 et 32
if (pin >= 0 && pin <= 32) {
// Extraire l'état de la sortie (dernier caractère)
bool state = (p->decoded.payload.bytes[11] == '1');

pinMode(pin, OUTPUT);
```

```
digitalWrite(pin, state);

// Affichage sur le port série pour debug
Serial.print("Commande reçue pour Node ");
Serial.print(nodeID);
Serial.print(" : Pin ");
Serial.print(pin);
Serial.print(" -> État ");
Serial.println(state ? "ON" : "OFF");

return true;
}
}
}
return false;
}
```

et NivekRemoteModule.h

```
#pragma once
#include "SinglePortModule.h"

class NivekRemoteModule : public SinglePortModule
{
public:
NivekRemoteModule() : SinglePortModule("nivekRemote",
meshtastic_PortNum_TEXT_MESSAGE_APP) {}

protected:
virtual bool wantPacket(const meshtastic_MeshPacket *p) override;
};
```

et donc dans scr/modules/Modules.cpp
ligne 145

```
new NivekRemoteModule();
```
ligne 93

```
#include "modules/NivekRemoteModule.h"
```